Abstract Presentation Back to Agenda Next Abstract

High-Performance Linear Algebra Processor using FPGA [*]

J. R. Johnson[†] P. Nagvajara[‡] C. Nwankpa[§]

1 Extended Abstract

With recent advances in FPGA (Field Programmable Gate Array) technology it is now feasible to use these devices to build special purpose processors for floating point intensive applications that arise in scientific computing. FPGA provides programmable hardware that can be used to design custom hardware without the high-cost of traditional hardware design. In this talk we discuss two multi-processor designs using FPGA for basic linear algebra computations such as matrix multiplication and LU factorization. The first design is a purely hardware solution for dense matrix computations, and the second design uses a hardware/software solution for sparse matrix computations. The hardware solution uses the regular structure available in dense linear algebra computations to design custom processors with hard-wired communication patterns. The hardware/software solution uses embedded processors with the flexibility to program the irregular communication patterns required by sparse matrix computations.

The dense matrix processor utilizes a distributed memory architecture connected in a ring topology, with hardwired control for communication. Each processing element consists of pipelined multiply-accumulate hardware, and local memory to store part of the input and output matrices.

[*]This work was partially supported by DOE grant #ER63384, PowerGrid - A Computation Engine for Large-Scale Electric Networks

[†]Department of Computer Science, Drexel University, Philadelphia, PA 19104. email: `jjohnson@cs.drexel.edu`

[‡]Department of Electrical and Computer Engineering, Drexel University, Philadelphia, PA 19104. email: `nagvajara@ece.drexel.edu`

[§]Department of Electrical and Computer Engineering, Drexel University, Philadelphia, PA 19104. email: `chika@nwankpa.ece.drexel.edu`

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **20 AUG 2004**	2. REPORT TYPE **N/A**	3. DATES COVERED **-**

4. TITLE AND SUBTITLE **High-Performance Linear Algebra Processor using FPGA**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Drexel University**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES
See also ADM001694, HPEC-6-Vol 1 ESC-TR-2003-081; High Performance Embedded Computing (HPEC) Workshop (7th)., The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**UU**	**19**	

Extra buffers are available to provide overlapped communication and computation so that while a computation is being performed the next inputs can be downloaded from the host computer. This allows the FPGA to be used as a hardware accelerator as part of a larger matrix computation using various block algorithms. In fact, the matrix processor supports BLAS routines such as GEMM (http://www.netlib.org/blas/).

The sparse matrix processor also uses a distributed memory architecture, however, it uses embedded processor cores which execute parallel programs written in C. A ring interconnect was designed and special instructions were added to the processor cores to support interprocessor communication. In addition, hardware support for floating point computations were added which are accessible to the processor through extra instructions. A parallel algorithm for LU factorization was implemented.

The resulting processor for sparse LU factorization is being designed for applications in power computations, where very large sparse systems arising from load-flow computation are solved as part of the Newton-Raphson method for solving a system of non-linear equations. These systems have problem specific structure which can be incorporated into the processor design. While the current investigation centers around this application, many other problems could benefit from a high-performance processor geared towards both sparse and dense matrix computations.

In this work parameterized designs, allowing a scalable number of processors, were created using the hardware definition language VHDL. The resulting design was synthesized and testing using the cyclone development board from Altera (http://www.altera.com/). Embedded NIOS processors were used for the processor for sparse LU factorization. While this board allowed verification of the design, it does not have sufficient hardware resources to adequately demonstrate the potential performance. Current FPGA devices have built-in hardware multipliers and large amounts of on chip memory, in addition to a large number of programmable logic blocks. These resources are crucial to obtaining high-performance, especially for computations with floating point numbers. For example, the Xilinx Virtex 2 (XC2V8000), see www.xilinx.com, has 168 pipelined 18-bit multipliers which can be used to implement 42 single-precision multiply-accumulate units.

Consequently a high-level performance model was created to estimate the performance using the high-end FPGA devices that are currently available and those that will be available in the next few years. The predicated performance was compared to efficient software solutions on high-speed workstations and clusters of workstations. Using clock speeds and the number of processors that can be implemented on the Virtex 2, a

400×400 matrix can be multiplied in 11ms, which corresponds to 11,570 MFLOPS. This compares to 10,000 MFLOPS obtained using a four processor 800 MHz Itanium processor with the Intel Math Kernel library (see http://www.intel.com/software/products/mkl/mkl52/specs.htm).

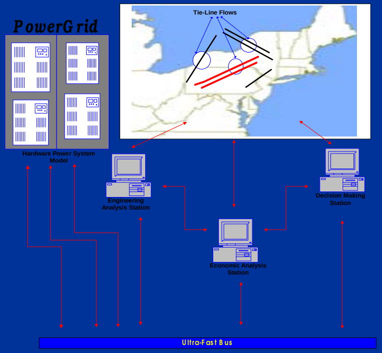
A similar performance model was used for the sparse linear solver. Assuming eight processors running at 400MHz with an 80 MHz floating point unit, which is currently possible using Xilinx FPGAs with a floating point core sold from Digital Core Design (http://www.dcd.com.pl/), the linear system needed to solve the 7917-bus power system could be solved in .084 seconds as compared to .666 seconds using the WSMP sparse linear solver (http://www.research.ibm.com/math/OpResearch/wsmp.html) on a 1.4GHz Pentium IV. While the times reported here are estimates based on a performance model, and hence should not be taken at face value, they show enough promise that the use of high-end FPGA devices, on a board with multiple FPGAs, to build a special-purpose linear algebra processor should be seriously investigated.

Abstract Presentation Back to Agenda Next Abstract

Linear Algebra Processor using FPGA

Jeremy Johnson, Prawat Nagvajara, Chika Nwankpa

Drexel University


PowerGrid
Tie-Line Flows
Hardware Power System Model
Engineering Analysis Station
Decision Making Station
Economic Analysis Station
Ultra-Fast Bus

Goal

- To design an embedded FPGA-based multiprocessor system to perform high speed Power Flow Analysis.
- To provide a single desktop environment to solve the entire package of Power Flow Problem (Multiprocessors on the Desktop).
- Provide a scalable solution to load flow computation.
- Deliver: Prototype and feasibility analysis.

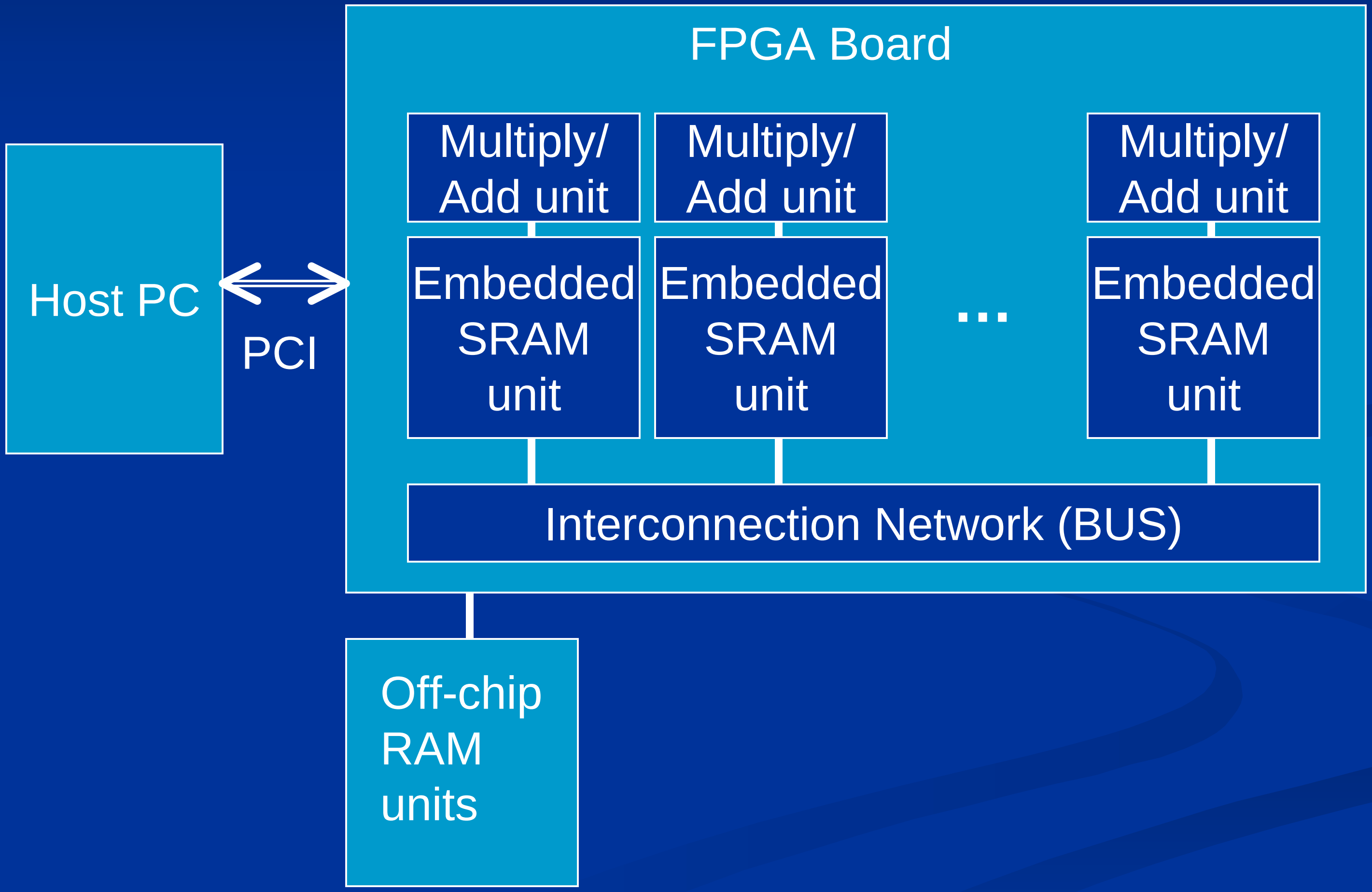
Approach

- Utilize parallel algorithms for matrix operations needed in load flow.
- Utilize sparsity structure across contingencies.
- Use multiple embedded processors with problem specific instruction and interconnect.
- Scalable parameterized design.
- Pipelined solution for contingency analysis.

Dense Matrix Multiplier

- Distributed memory implementation
  - Hardwired control for communication
  - Parameterized number of processing elements (multiply and accumulate hardware)
  - Overlap computation and communication
  - Use block matrix algorithms with calls to FPGA for large problems
- Processor i stores A_i, the ith block of rows of A and B_j, the jth block of columns of B
  - Compute $C_{ij} = A_i * B_j$
  - Rotate: send B_j to processor (j+1) mod Number of processors

Processor Architecture



Performance

- Performance estimate
  - Xilinx Virtex 2 (XC2V8000)
- 168 built-in multipliers and on chip memories (SRAM) $\Rightarrow$ support for 42 single-precision processing elements
  - 4.11 ns pipelined 18 X 18 bit multiplier
  - 2.39 ns memory access time
  - 7.26 ns for multiply accumulate

Time for $n \times n$ matrix multiply with p processors
  - $7.26n^3/p$ ns
  - 11 ms for n=400, p = 42 $\Rightarrow$ 11,570 MFLOPS

APPLICATION, ALGORITHM & SCHEDULE

- Application
  - Linear solver in Power-flow solution for large systems
  - Newton-Raphson loops for convergence, Jacobian matrix
- Algorithm & Schedule
  - Pre-permute columns
  - Sparse LU factorization, Forward and Backward Substitutions
  - Schedule: Round Robin distribution of rows of Jacobian matrix according to the pattern of column permutation

Breakdown of Power-Flow Solution Implementation in Hardware

Problem

Formulation

(Serial)

Problem

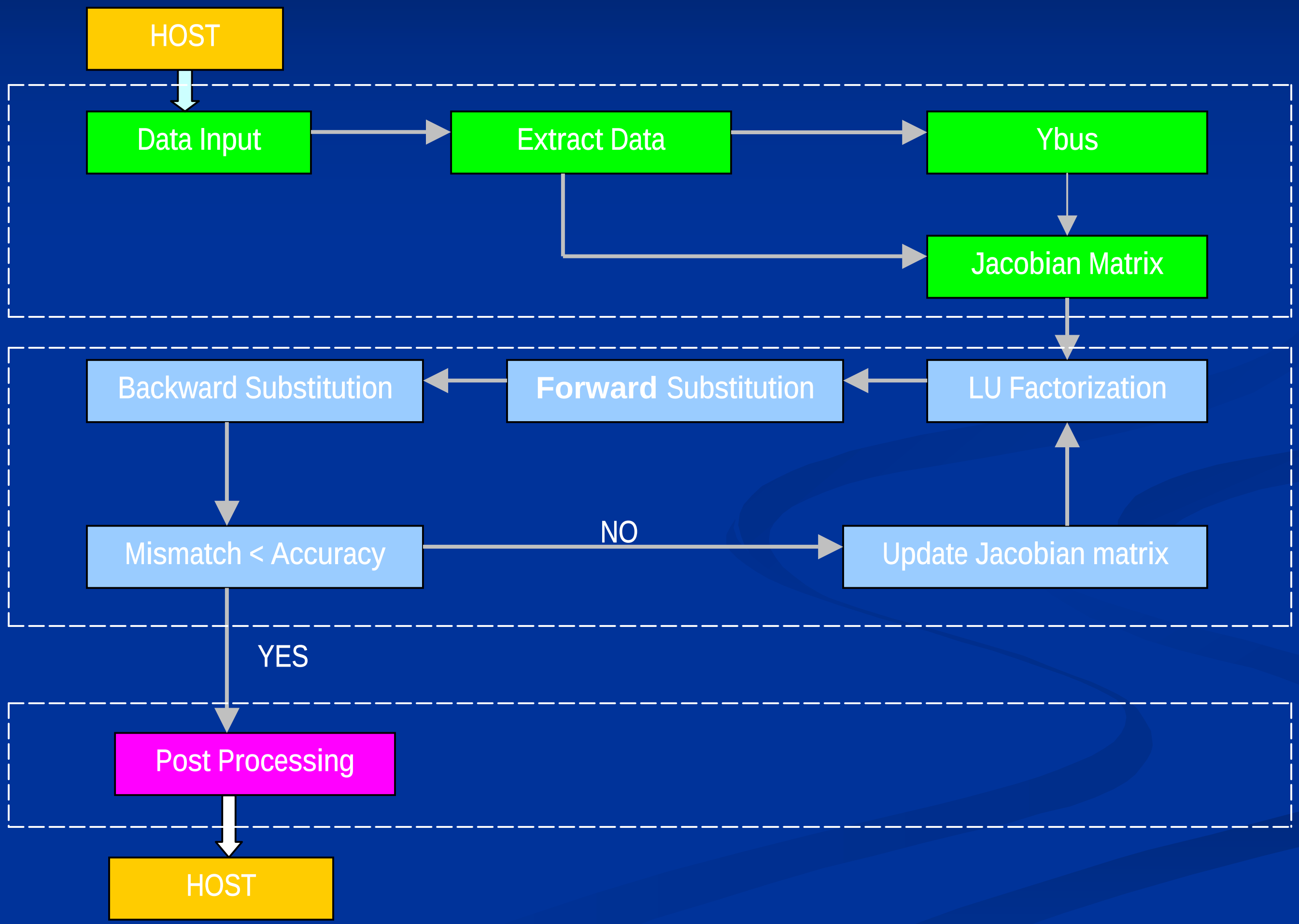
Solution

(Parallel)

Post

Processing

(Serial)

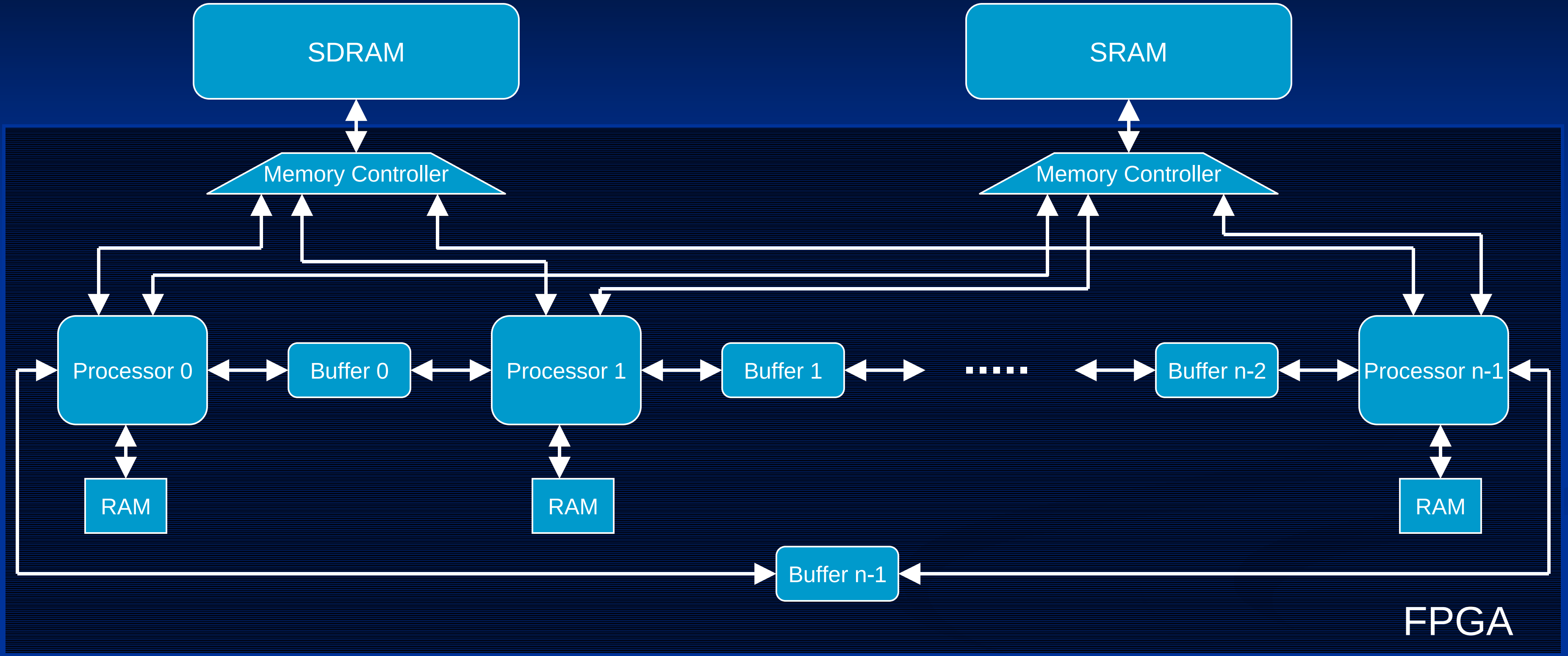


Minimum-Degree Ordering Algorithm (MMD)

- Reordering columns to reduce fill-ins while performing LU factorization
- Reduce floating point operations and storage
- Compute column permutation pattern once
- Apply throughout power-flow analysis for that set of input bus data

	Without MMD			With MMD		
	Div	**Mult**	**Sub**	**Div**	**Mult**	**Sub**
IEEE 30-bus	887	25806	25806	270	6535	6535
IEEE 118-bus	4509	340095	340095	719	55280	55280
IEEE 300-bus	31596	5429438	5429438	3058	640415	640415

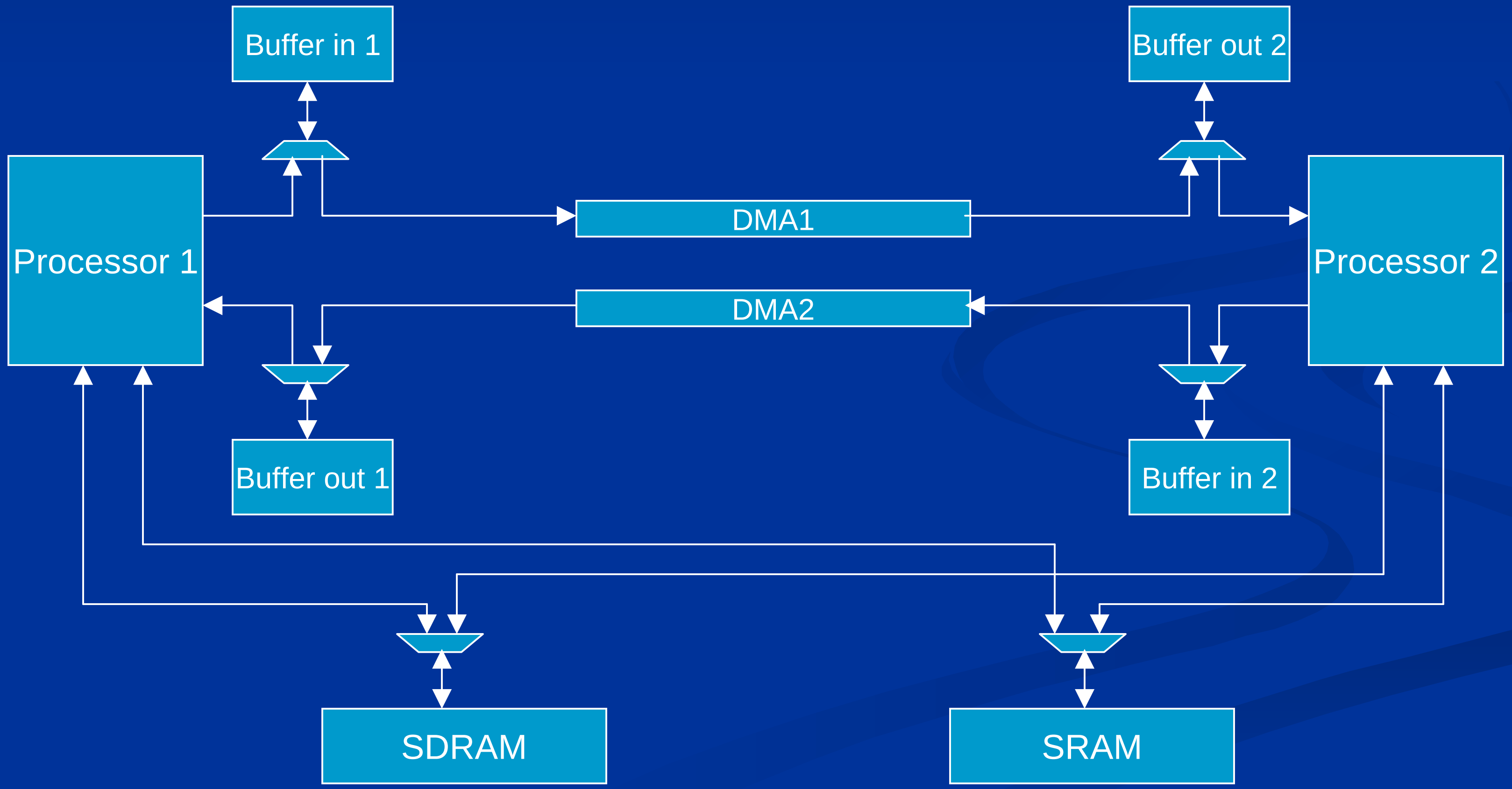
RING TOPOLOGY



- Nios Embedded Processors
- Buffers are on-chip memories; interprocessor communication

Hardware Model using Nios Processor

- Communication: used DMA for passing messages
  - Buffers are on-chip memories
  - Trapezoids are arbitrators

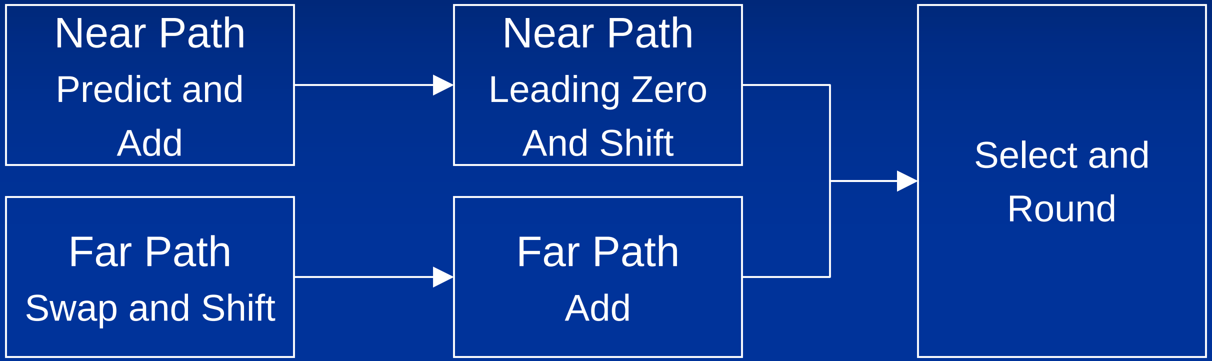


Stratix FPGA

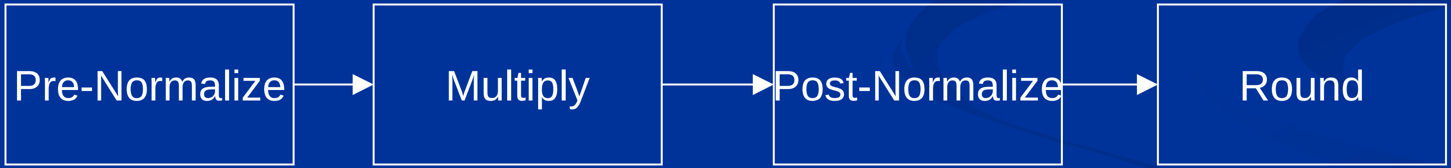
Stratix 1S25F780C6 Resources	
LEs	25,660
M512 RAM blocks (32 x 18 bits)	224
M4K RAM blocks (128 x 36 bits)	138
M-RAM blocks (4K x 144 bits)	2
Total RAM bits	1,944,576
DSP Blocks	10
Embedded multipliers	80
PLLs	6

Floating Point Unit

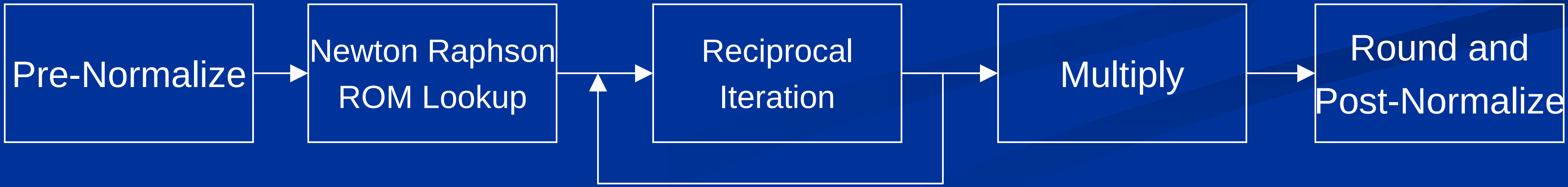
FADD



FMUL



FDIV



Floating Point Unit

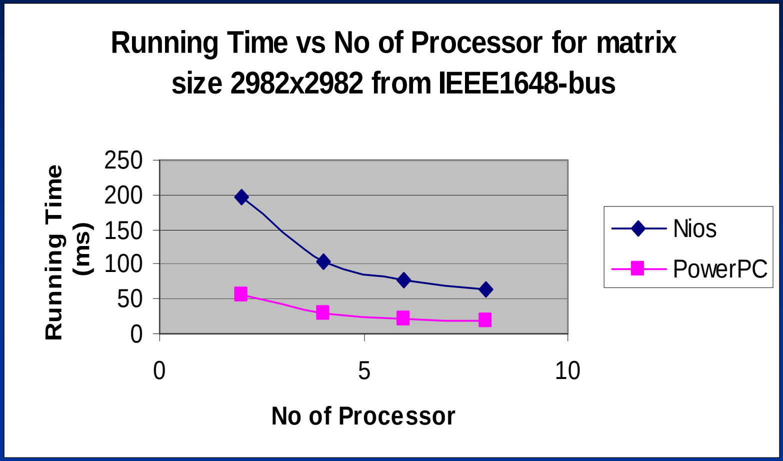
- IEEE-754 Support
  - Single Precision Format, Round to Nearest
  - Round Nearest Rounding, Denormalized Numbers, Special Numbers

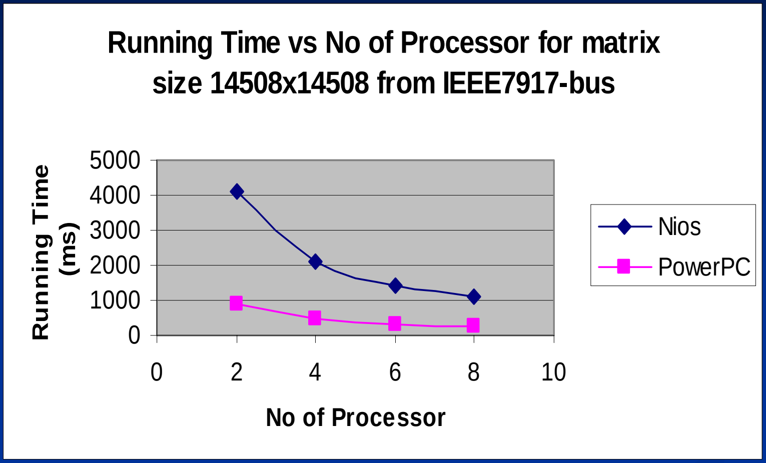
	LEs	MULs	Fmax (MHz)	Latency (clk cycles)	Pipeline Rate (clk cycles)
FADD	1088	0	91	3	1
FMUL	926	8	107	4	1
FDIV	1166	8	65	9	N/A
NIOS + FPU	6134		63.88		

PERFORMANCE ANALYSIS

- Why?
  - Prototype is not intended for industrial performance
  - Show potential performance as a function of available H/W resources
- Analysis performed for high-performance multi-processor system
  - Estimate of number of clock cycles and timing
    - Communication latency
    - Arithmetic latency
  - Model in MATLAB
  - 80 MHz pipelined Floating-point Unit
  - Variables
    - Number of Processors (2, 4, 6, 8)
    - Size of input data (power flow IEEE1648-bus, IEEE7917-bus)
  - System constraints: memory access, size of FPGA chip, system speed

TIMING OF PERFORMANCE MODEL





•Nios embedded processors on Altera Stratix FPGA, running at 80 MHz

- 8 processors:
  - 1648-bus: 64.179 ms
  - 7917-bus: 1106.676 ms

• 400 MHz PowerPC on Xilinx Virtex 2 FPGA with 80 MHz FPU

- 8 processors:
  - 1648-bus: 18.992 ms
  - 7917-bus: 256.382 ms

• WSMP – 1.4 GHz, 256 KB Cache, 256 MB SDRAM, Linux OS

- 1648-bus: 146.435 ms
- 7917-bus: 666.487 ms